маркетингового завдання розробляється відповідний інструментарій і конкретні заходи, спрямовані на обрані сегменти, та розподіляються у часі згідно з календарним планом будівництва.

Грамотно розроблений дієвий комплекс маркетингових комунікацій, що здійснює безперервний вплив на цільову аудиторію від підготовки будівництва і до післяпродажного обслуговування, дозволить збільшити обсяги продаж, змістити продажі на більш ранні стадії будівництва, підвищити лояльність споживачів, забезпечити встановлення тісних зв`язків з учасниками ринкової діяльності.

Література

1. Барнет Дж. Маркетинговые коммуникации. Интегрированный подход / Дж. Барнет, С. Мариарти ; пер. с англ. – СПб., 2001. – 864 с.

2. Васильченко Л. С. Управління системою маркетингових комунікацій / Л. С. Васильченко, Т. І. Бурцева // Східна Европа: економіка, бізнес та управління. – 2016. – Вип. 2 (02). – С. 145–148.

3. Ухова А. И. Развитие подходов к формированию комплекса маркетинговых коммуникаций на рынке первичной жилой недвижимости / А. И. Ухова, В. М. Каточков // Современные проблемы науки и образования. – 2015. – № 1-1.

4. Обсяги будівництва. Аналітика [Електронний ресурс]. – Режим доступу: budport.com.ua.

МЕНЕДЖМЕНТ У СФЕРІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кравчук О. А. Хмельницький національний університет, Україна
E-mail:kravchukoa2@gmail.com

Менеджмент у сфері програмного забезпечення (англ. Software product management) являє собою сукупність процесів управління даним продуктом на різних його стадіях, тобто на різних етапах життєвого циклу товару, і з врахуванням інтересів користувачів продукту. Це наука та бізнес-процес водночас, як правильно здійснювати управління у галузі від початку виникнення потреби у продукті і до постачання його на ринок, та обслуговування клієнтів з метою отримання максимальної цінності та значення для бізнесу [3].

Розробка програмного забезпечення (ПО) в більшості випадків розглядається як колективна праця фахівців, спрямований на задоволення потреби користувачів в автоматизації їх діяльності. Як і будь-яка інша колективна праця, вона вимагає організації, зокрема, управління. Це процес часом тривалий, що зв'язаний виробничими та ін-

шими відносинами тих, кого в тій чи іншій мірі можна розглядати в якості виробників програми.

Найбільш типову модель процесу виробництва програмного забезпечення можна охарактеризувати наступним чином: «Кожен розробник вибирає той чи інший метод або техніку для створення програм відповідно до власних звичками та вподобаннями. Практично повна відсутність чіткої відповідальності за виконання тих чи інших функцій. Якість програмного забезпечення є випадковою величиною і безпосередньо залежить від здібностей окремих співробітників компанії. Практично все залежить від ініціативи та ділових якостей декількох особистостей». Це формулювання практично повністю відповідає першому рівню CMM під назвою «початковий». За деякими джерелом, два роки тому частка фірм-виробників програмного забезпечення, що використовують цю модель, становила понад 70 %.

Ось які висновки робить Мартін Фаулер [1], порівнюючи процес виробництва програмного забезпечення з класичними типами будівництва та виробництва:

– в процесі виробництва ПО фаза безпосереднього програмування (construction) набагато дешевше всіх інших фаз (проектування, тестування і т.п.);

– в процесі виробництва ПО всі основні зусилля спрямовуються на проектування. Процес проектування вимагає, щоб в ньому брали участь творчі та обдаровані люди;

– творчі процеси не можна легко запланувати. Таким чином, передбачуваність подібних процесів може бути недосяжною метою;

– ми повинні дуже обережно ставитися до використання традиційних метафор при виробництві програмного забезпечення. Це абсолютно особливий вид діяльності, що вимагає особливого процесу.

Класичний процес виробництва програмного забезпечення, який використовувався у всьому світі і який практично є символом епохи структурного програмування, складається з наступних кроків: обстеження, постановка задачі, проектування, програмування, тестування і впровадження. Цей процес називається «Водоспад». Він має на увазі, що вимоги до програмного продукту, зібрані під час обстеження та формалізовані в процесі постановки завдання, зафіксовані і не змінюються протягом всього виробничого циклу. Однак, сучасний бізнес дуже динамічний і зміна вимог в ньому – звичайна справа.

Фаулер пише: «В процесі виробництва програмного забезпечення все залежить від вимог. Якщо ви не можете добитися стійкості вимог, ви не зможете створити передбачуваний план ». Як же бути? З одного боку вимоги повинні бути стійкими, а з іншого вони будуть неминуче змінюватися в ході проекту.

Всі шляхи вирішення вищеозначених проблем зводяться до зміни процесу виробництва програмного забезпечення таким чином, щоб процес був передбачуваним, стійким і вчасно забезпечував би виконання головної мети – готове програмне забезпечення.

Алістер Кобурн, автор Crystal, є одним з найбільш відомих сучасних фахівців, які зробили об'єктом своїх досліджень методології. Результатом його досліджень в даній області є книга «Agile Software Development» [4], в якій автор аналізує процес виробництва ПО з різних точок зору. Алістер Кобурн є автором концепції «Розробка програмного забезпечення – це спільна гра винаходи і взаємодії». Він виділяє дві мети у цю гру:

1) розробити і впровадити працюче програмне забезпечення;

2) забезпечити всі умови для успішного перебігу наступної гри.

Таким чином, формулюється мінімаксове завдання: привести на світ ПО (найчастіше для цього не потрібно створювати взагалі ніяких документів) і забезпечити розвиток наступних версій даного ПЗ (а ось для цього необхідно створювати ряд документів, які необхідні, наприклад, в разі, якщо один з основних розробників покине команду). У той же час, легкі процеси не застосовні для роботи на великих проектах або над продуктами, ступінь критичності правильної роботи яких дуже висока.

Література

1. Martin Fowler. The New Methodology (www.martinfowler.com/articles/newMethodology.html)

2. Spice. Consolidated product. Software Process Assessment (http://www.sqi.gu.edu.au/spice)

3. https://uk.wikipedia.org/wiki

4. Cockburn A. Agile Software Development, 2001, Addison Wesley.